KOLEJ PROFESIONAL MARA BERANANG

DIPLOMA IN COMPUTER SCIENCE

COURSE NAME	:	**OBJECT ORIENTED PROGRAMMING**
COURSE CODE	:	**CSC2744**
ACADEMIC SESSION	:	**SESSION 1 2023/2024**
TYPE OF ASSESSMENT	:	**FINAL ASSIGNMENT**
DURATION	:	**20/6/2023-10/07/2023**

CLO 3: Employ third party data in object oriented application development using graphical user interface (GUI) application framework

INSTRUCTION TO CANDIDATES:

1. Late submissions after given due date will not be accepted.
2. Report should be written using:
   Font type: Arial
   Size: 12 pts
   Line Spacing: 1.5
3. Coding format:
   Font type: Consolas
   Size: 10 pts
   Line Spacing: Single

Personal Details	
Name	AIN ZAHIRAH BT AINUDDIN
I/D Number	BCS2207-009
Class	DCS 4A
Lecturer	PUAN NINI ANIZA

Section / Question No.	**Marks**
Total	**/ 50**

Question

Electron is a powerful framework that enables developers to create cross-platform applications using web technologies such as HTML, CSS, and JavaScript. You need to choose one of the applications below to develop a desktop application using Electron that integrates the given API. The application needs to be developed with specific requirements or functionalities.

Name of Application	Description	Requirements
Dictionary and Thesaurus	An app that lists words in groups of synonyms and related concept.	• Word search. • Information Output: give the meaning of the searched word for different part of speech (noun/adjective), antonyms and the example of word usage, sounds and related URL for the searched word. • CRUD words of the day https://api.dictionaryapi.dev/api/v2/entries/en/digital
Meal planner	An app that displays suggestion of recipe based on food item entered by user	• Suggest recipe based on food item. • Information Output: One recipe suggestion that comprises of ingredients, instruction on how to cook and URL of the related site for the food and the link on how to prepare the food. • CRUD meal planner https://www.themealdb.com/api/json/v1/1/search.php?s=shawarma
Makeup Box	An app that finds makeup products based on brand and category entered.	• Display the product info according to search criteria. • Information Output: product description based on brand and name, product image, product website and related link for the searched product. • CRUD makeup top 5 list http://makeupapi.herokuapp.com/api/v1/products.json?brand=maybelline

Tasks:

1. Create desktop app using electron and apply third party data fetched from API and the requirements given. Your application should have at least 2 pages and you may add extra functionality or features of your choice to the application.
2. Implement CRUD (create, read, update, delete) process to the application as given in the requirements.
3. Apply HTML and CSS for user interface and provide evidence for application.
4. GUI Elements:
    i. Apply GUI elements that assist users in using application.
    ii. The application's 'look and feel' is attractive and informative.
5. Produce a report on your application functionalities and features. Include the following:
    i. Overview of your application with a brief description.
    ii. Screenshots of the application with explanations on how to use it.
    iii. Program codes of your system
6. Submit files in GitHub.

```
ipcMain.on("item:add", function(e,item){
  mainWindow.webContents.send("item:add", item)
  addWindow.close()
 })
 //Create an array of menus

 var mainMenuTemplate = [
  {
   label:'File',
   submenu:[
    {
     label: 'Add new item',
     click(){
        addWindow = new BrowserWindow({
        width: 800,
        height: 600,
        title:"Add new item",
```

```
      webPreferences:{
        nodeIntegration: true,
        contextIsolation: false
      }

    });

    // and load the index.html of the app.
    addWindow.loadFile(path.join(__dirname, 'add.html'));
  }
},
```

Assessment Rubric

ATTRIBUTES	CRITERIA	POOR (1 mark)	FAIR (2 marks)	GOOD (3 marks)	VERY GOOD (4 marks)	EXCELLENT (5 marks)	Mark Obtained
Reproduce and Process Information	1. Create desktop app using electron and apply third party data fetched from API and the requirements given. You may add extra functionality or features of your choice to the application.	The application is an extensive collection and rehash of other people's ideas, products, and images. There is no evidence of new thought.	The application is somewhat a collection and rehash of other people's ideas, products, and images. There is little evidence of new thought or inventiveness.	The application is a minimal collection or rehash of other people's ideas, products, and images. There is a few evidence of new thought or inventiveness.	The application shows a lot of evidence of originality and inventiveness.	The application shows significant evidence of originality and inventiveness. Most of the content and many of the ideas are fresh,original, and inventive.	
		Able to display part of the data from the API and does not fulfill the requirements.	Able to display sufficient data from the API that meet with the requirements together with the description.	Able to display sufficient data from the API that meet with the requirements together with the description.	Able to display extra data from the API beyond the application requirements together with no description.	Able to display extra data from the API beyond the application requirements together with the description.	
		The data from the API does not reflect the whole purpose of the application developed.	The data from the API is sufficient but does not reflect the whole purpose of the application developed.	The data from the API is meaningful but does not reflect the purpose of the application developed.	The data from the API is meaningful and reflect the purpose of the application developed.	The data from the API is meaningful to come up with extra idea for the application developed.	
		The developed application does not fulfill the requirements stated for the chosen	The developed application fulfills all the requirements stated for the chosen application with	The developed application fulfills all the requirements stated for the chosen application.	The developed application fulfills all the requirements stated for the chosen application.	The developed application fulfills all the requirements stated for the chosen application that	

		application.	no extra functionalities.	Add extra functionality or features to the application.	Add extra functionality or features to the application that enhances the user experience or adds value to the application.	utilizes the data from the API Add extra functionality or features to the application that enhances the user experience or adds value to the application.	
	2. Implement CRUD (create, read, update, delete) process to the application as given in the requirements.	• Able to create only 2 of the CRUD processes according to the requirements. • No feedback for the CRUD process. • The design for data input is poor	• Able to create only 3 of the CRUD processes according to the requirements. • No feedback for the CRUD process. • The design for data input is good with some room for improvement	• Able to perform all the CRUD processes according to the requirements. • No feedback for the CRUD process. • Well-designed data input for CRUD process.	• Able to perform all the CRUD processes according to the requirements. • No feedback for the CRUD process. • Well-designed data input for CRUD process.	• Able to perform all the CRUD processes according to the requirements. • Appropriate feedback for the CRUD process. • Well-designed and user-friendly data input for CRUD process.	
	3. Apply HTML and CSS for user interface and provide evidence for application.	• **Text -** All text used is too small to view or the font type is wrongly chosen. • **Graphics** - Graphics seem randomly chosen, are of low quality, OR distract the reader.	• **Text** – Some of the text used is too small to view or the font type is wrongly chosen. • **Graphics -** Graphics seem randomly chosen, are	• **Text** - Most text used is clear but does not describe the content well. • **Graphics** - Graphics are related to the theme/purpose of the application	• **Text** - All text used is clear but does not describe the content well. • **Graphics -** Graphics are related to the theme/purpose of the	• **Text** - All text used is clear and able to describe the content well. • **Graphics** - Graphics are related to the theme/purpose of the application, are thoughtfully	

			of low quality, OR distract the reader.	and are of excellent quality.	application, are of excellent quality and enhance reader interest or understanding	cropped, are of high quality and enhance reader interest or understanding.	
Curate	4.GUI Elements: i. Apply GUI elements that assist users in using application. i.	Not able to curate for required content. • **Layout** - The HTML elements in the application are cluttered looking or confusing. • **Navigation** Links do not take the reader to the sites/ pages described. User typically feels lost.	Limited curation for required content. • **Layout** - The HTML elements in the application is messy, may appear busy or boring. • **Navigation** Links seem to be missing and don't allow the user to easily navigate.	Satisfactory curation for required content. • **Layout** - The HTML elements are suitable. • **Navigation** Links allow the reader to move from page to page, but some links seem to be missing.	Good curation for required content. • **Layout** - The HTML elements are suitable and usable. • **Navigation** Links are labelled and allow the user to easily move from page to page.	Excellent curation for required content. • **Layout** - The HTML elements are well structured, attractive, and usable layout. • **Navigation** Links are clearly labelled, consistently placed, and allow the user to easily move from page to page.	
	ii. The application's 'look and feel' is attractive and informative.	• The application is in need of polish in its visual design and is not appropriate for the target audience. • **Color**	• The application is in need of polish in its visual design, but it is still appropriate for the target audience. • **Color**	• The application mostly follows good visual design principles (e.g.: alignment, contrast,	• The application demonstrates good visual design principles (e.g.: alignment, contrast,	• The application clearly demonstrates good visual design principles (e.g.: alignment,	

		Choice of colors and combinations are not suitable.	Choice of colors and combinations do not match the concept of the application.	easily read text) and is appropriate for the target audience. • **Color** Choice of colors and combinations match the concept of the application.	easily read text) and is appropriate for the target audience. • **Color** Appropriate colors used to produce an atmosphere that expresses the concept of the application.	contrast, easily read text) and is appropriate for the target audience. • **Color** Appropriate colors used to produce an atmosphere that expresses the concept of the application.	
Convey	5. Produce a report on your application functionalities and features that includes: i. Overview of the application. ii. Screenshots of the application with explanations on how to use it.	The overview of the application is vague. The user guide is incomplete and cannot be recognized as a user guide.	The overview of the application is very brief and does not describe the whole functionalities of the application. The user guide provides limited information with no screenshots of the application.	The overview of the application is clearly described the whole application and its functionalities. The user guide provides basic information with limited screenshots of the application.	The overview of the application is clearly described the whole application and its functionalities. The user guide provides adequate information with complete screenshots of the application.	The overview of the application is clearly described the whole application and its functionalities. The user guide provides extensive information with complete screenshots and labelling of the application.	
	iii. Program codes of the system	HTML, CSS and JavaScript codes attached are not complete. The codes are hardly read.	HTML, CSS and JavaScript codes attached are complete. The codes are hardly read.	HTML, CSS and JavaScript codes attached are complete. The codes are readable but not organized.	HTML, CSS and JavaScript codes attached are complete. The codes are readable and	HTML, CSS and JavaScript codes attached are complete and include comments for the important parts of the codes.	

		Does not submit complete electron files in GitHub	Completely submit all the electron file in GitHub.	Completely submit all the electron file in GitHub.	organized. Completely submit all the electron file in GitHub.	The codes are readable and organized. Completely submit all the electron file in GitHub.	
Total Marks Earned							**/50**
Total Percentage (40%)							**/40%**

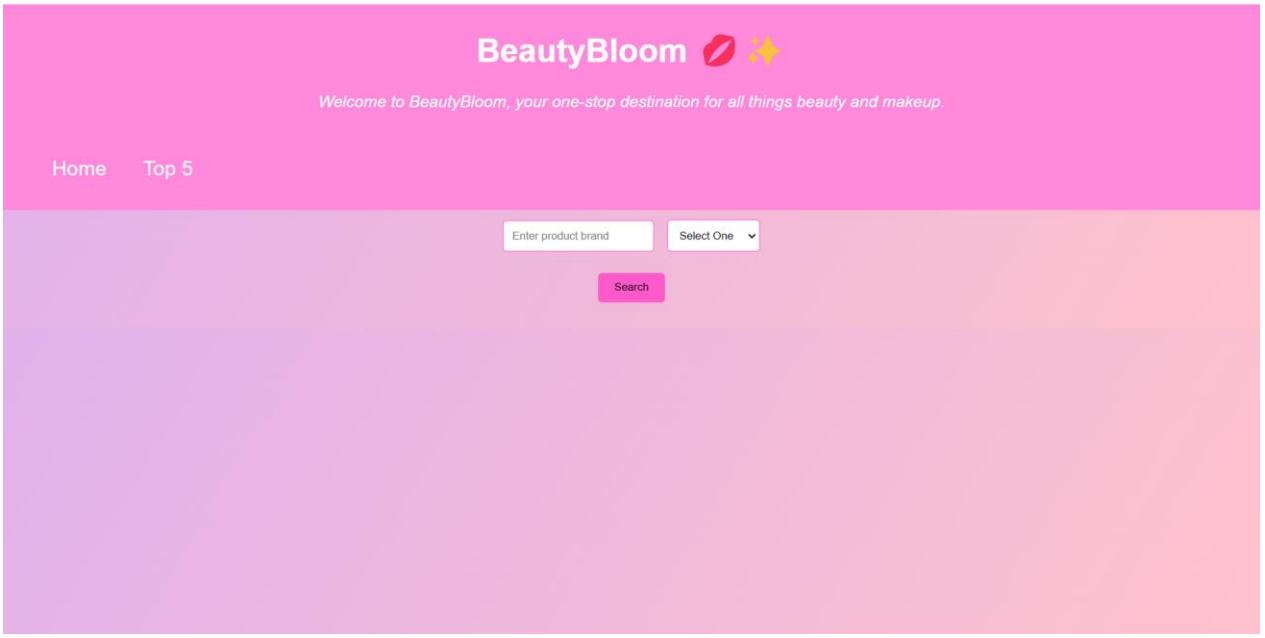
Overview

A makeup website is a digital platform dedicated to all aspects of makeup and beauty. These websites serve as valuable resources and destinations for makeup enthusiasts, professionals, and anyone interested in cosmetics and beauty-related content. For this BeautyBloom website, it provide user to search for makeup products based on product brands and product type. User can view product image, product name, product type, product description and product price on this website. User can also clicks on the product link and it will direct user to the official website of the product. In this website, user can also create top 5 makeup list by using on the top 5 page. On the page, user can create, read , delete and update the top 5 makeup list by using all the button and functions provided on the website.

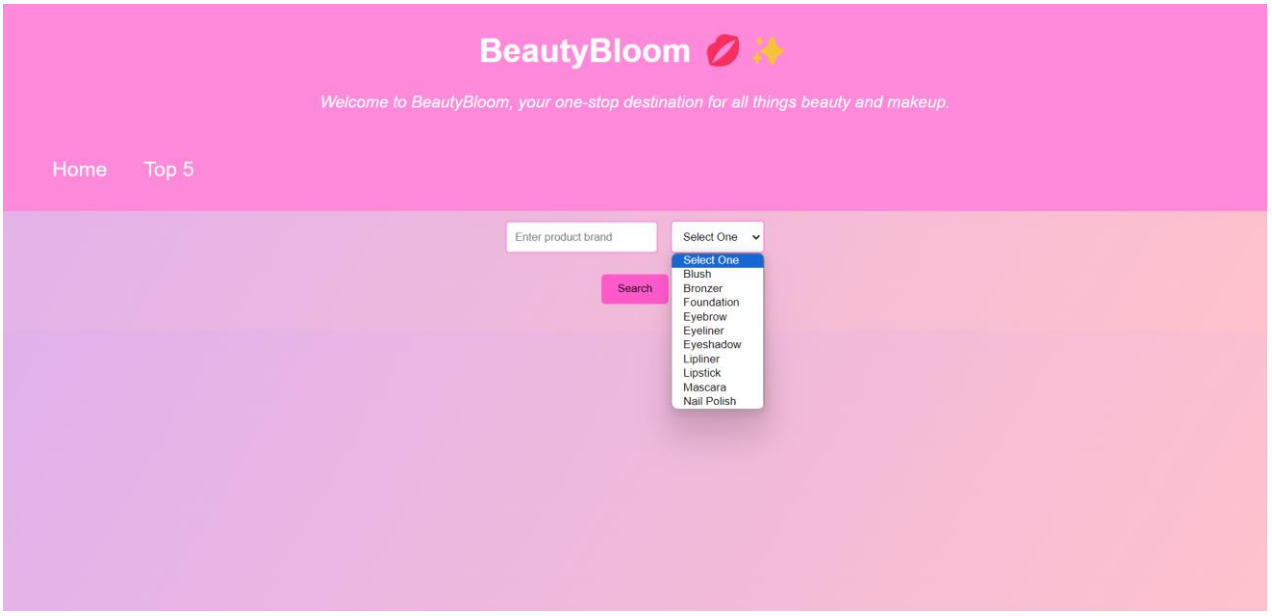
User Guide

Fetchapi.html

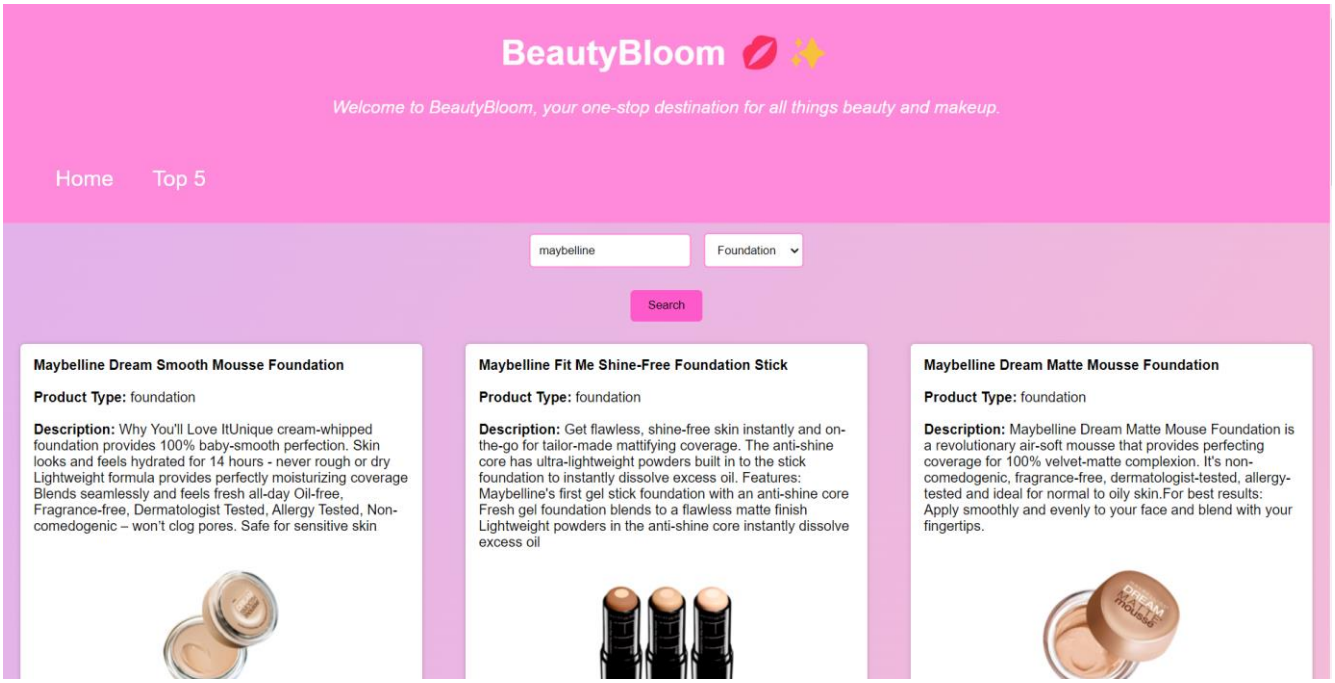
This is the homepage for the website. User can search for makeup products by entering makeup brands and select the product type on the dropdown. In this page, it has navigation bar that direct user to the other pages. When user click on home navigation bar, it will stay on the homepage and if user clicks on the Top 5 navigation bar, it will direct user to a new page where user can create file, read , update and delete the data about top 5 makeup list. When user clicks on the search button, it will display the data using the fetchapi method.



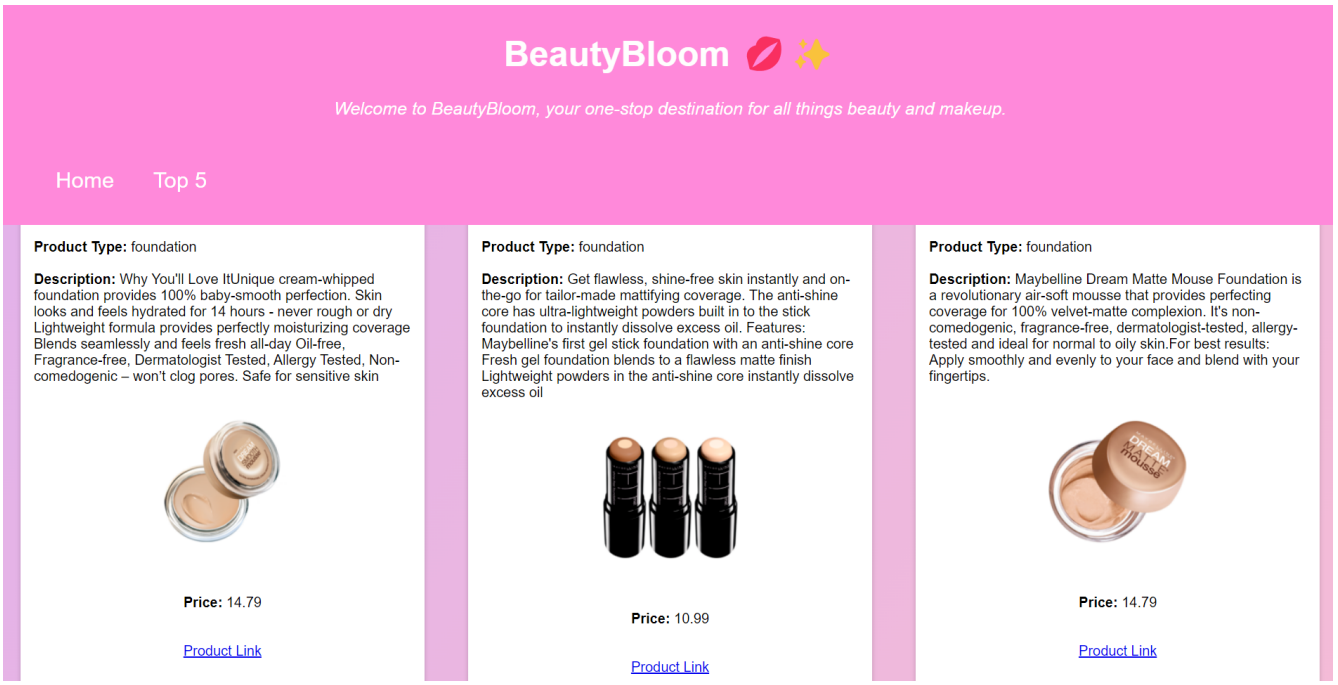
User clicks on dropdown button

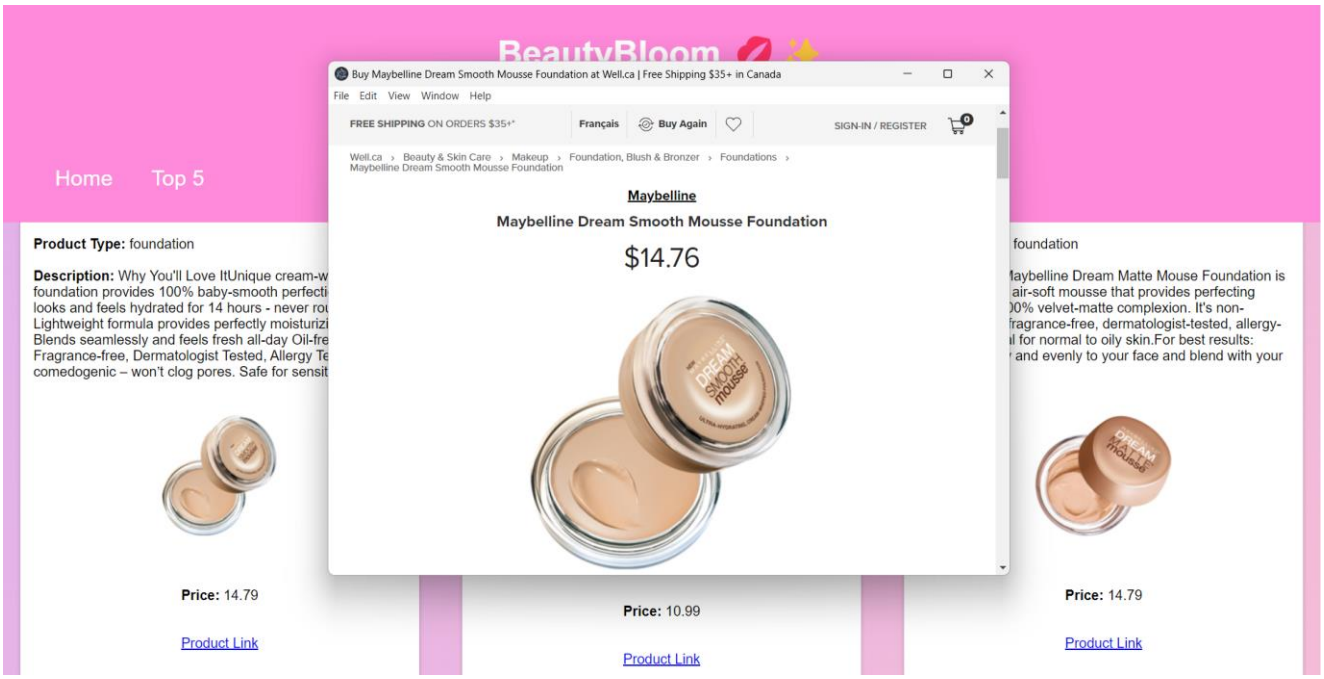


User clicks on search button

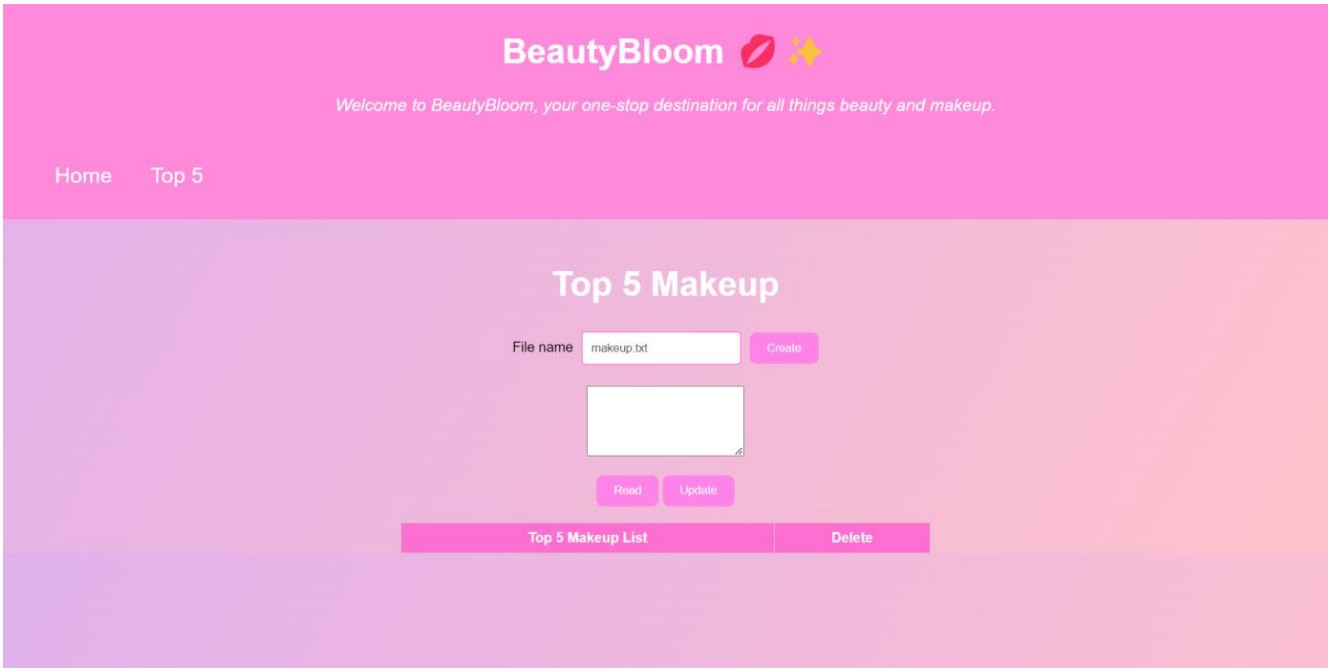


User clicks on product link



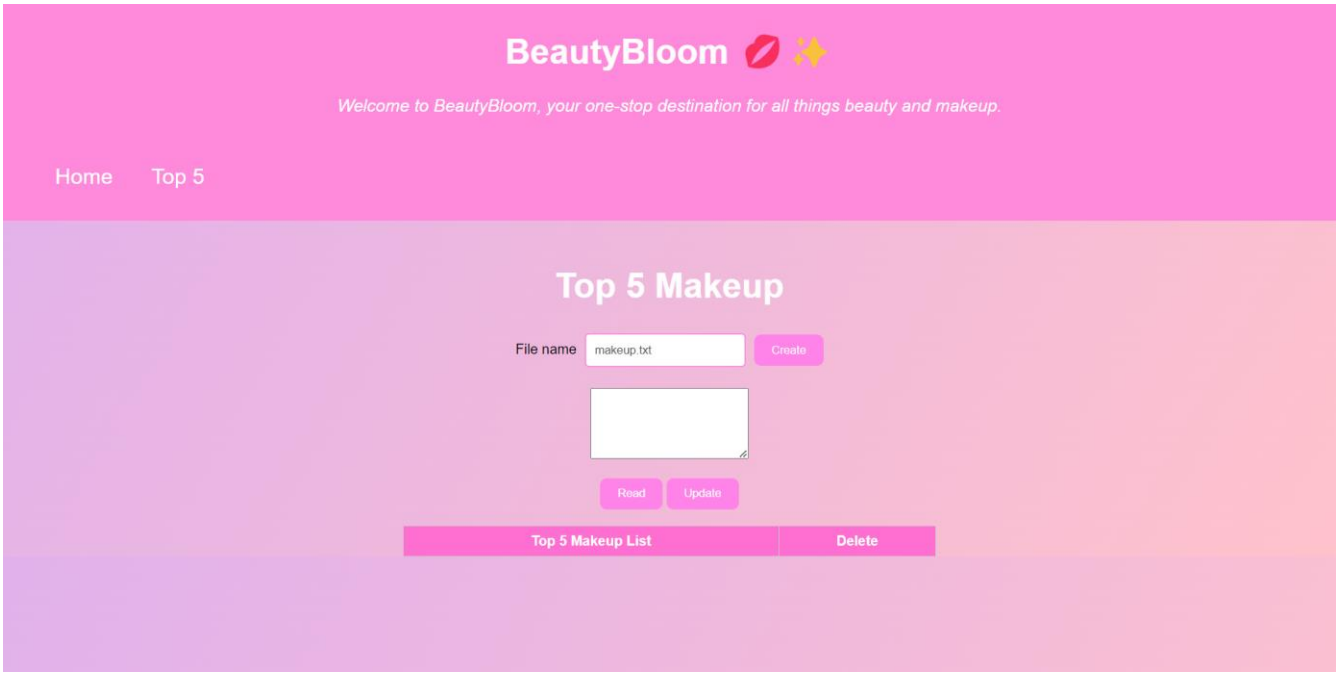


User clicks Top 5 navigation bar

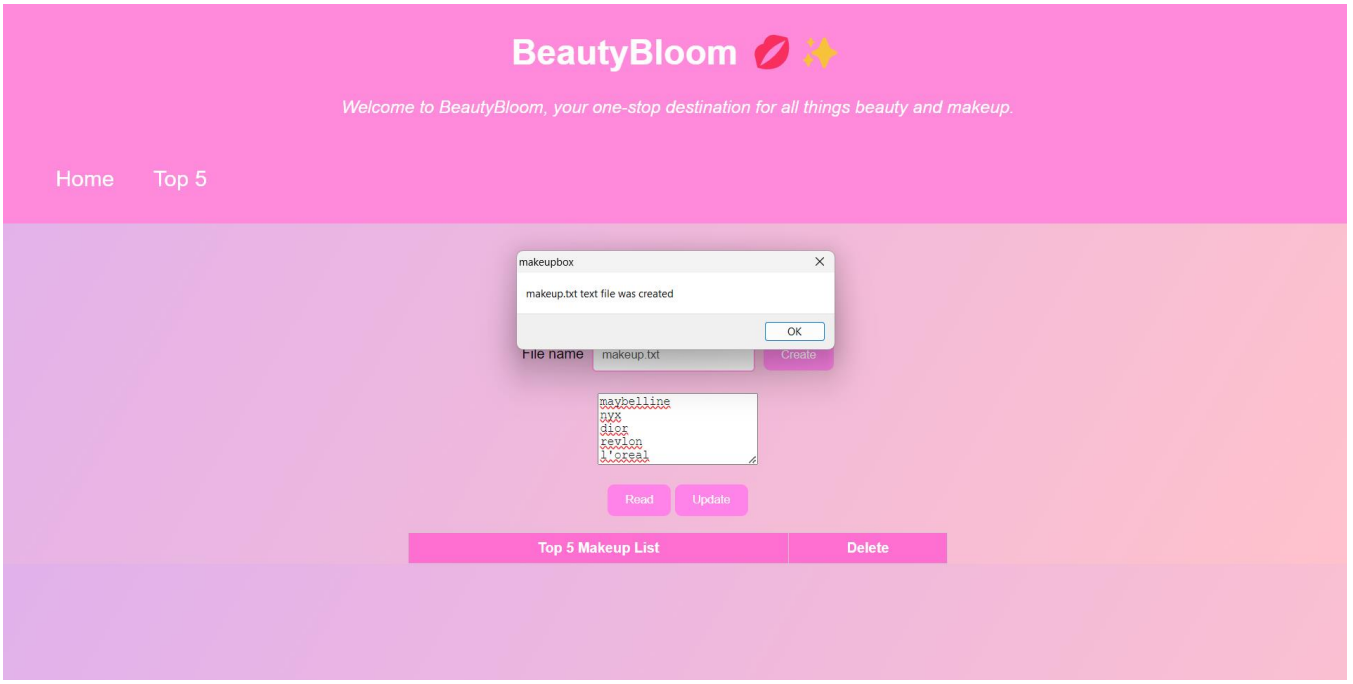


Crud.html

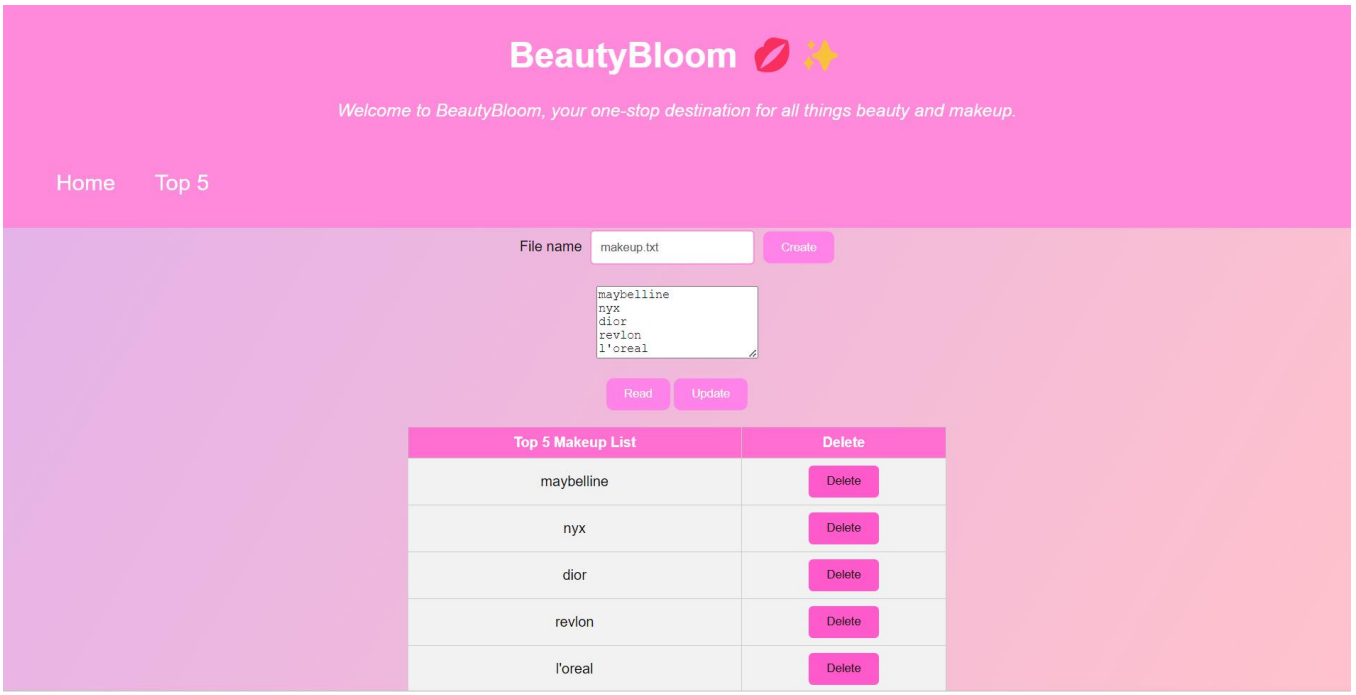
This is the page where user can create file, update, read and delete data in using all the button provided in this page. In this page, it has navigation bar that can direct user to the other pages. When user click on home navigation bar, it will direct user to the homepage and if user clicks on the Top 5 navigation bar, it will stay on the page. The file name is fixed so that user can create the top 5 makeup list in the same file. After user created top 5 makeup list, the website will display a popup telling user the file has been created. User can also click on the read button to read the data in the file. User can also click on the update button to update data in the file. After user click on the read button, the website will display the top 5 makeup list in the Top 5 Makeup List table below. In this page, user can also delete the makeup list when they click on the delete button to delete the selected makeup list.



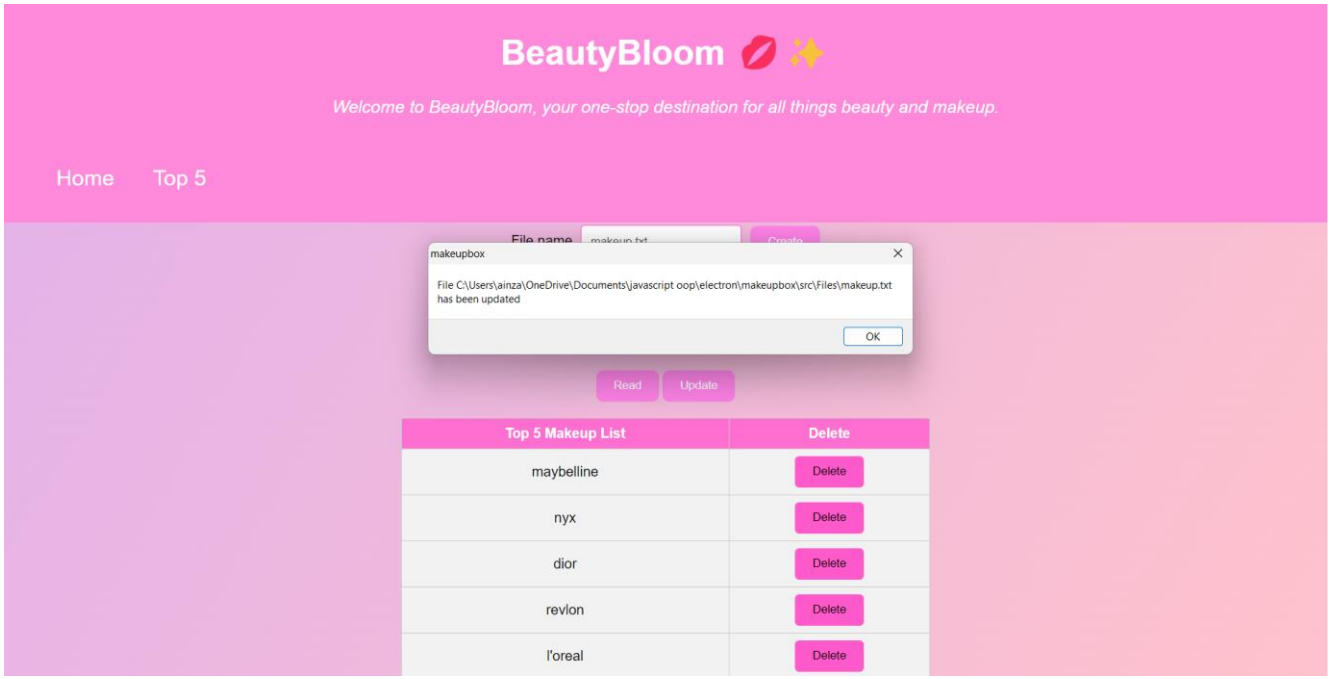
User clicks on the create button

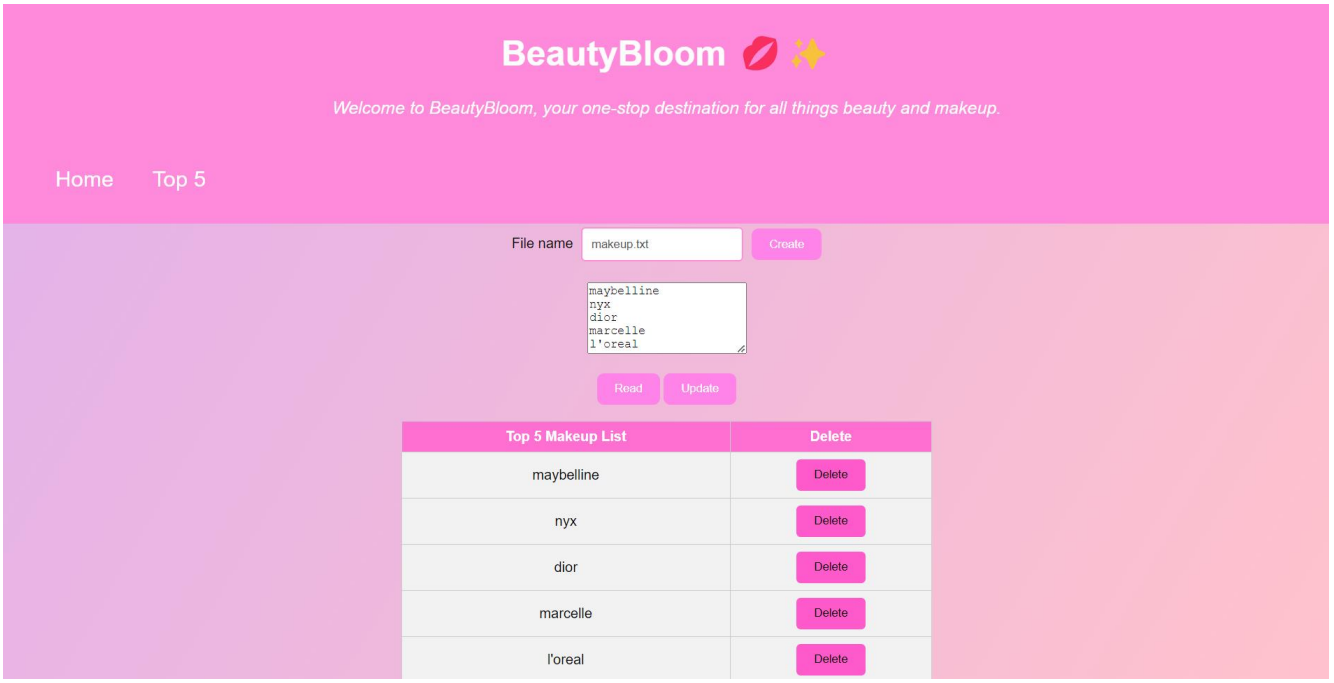


User clicks on the read button

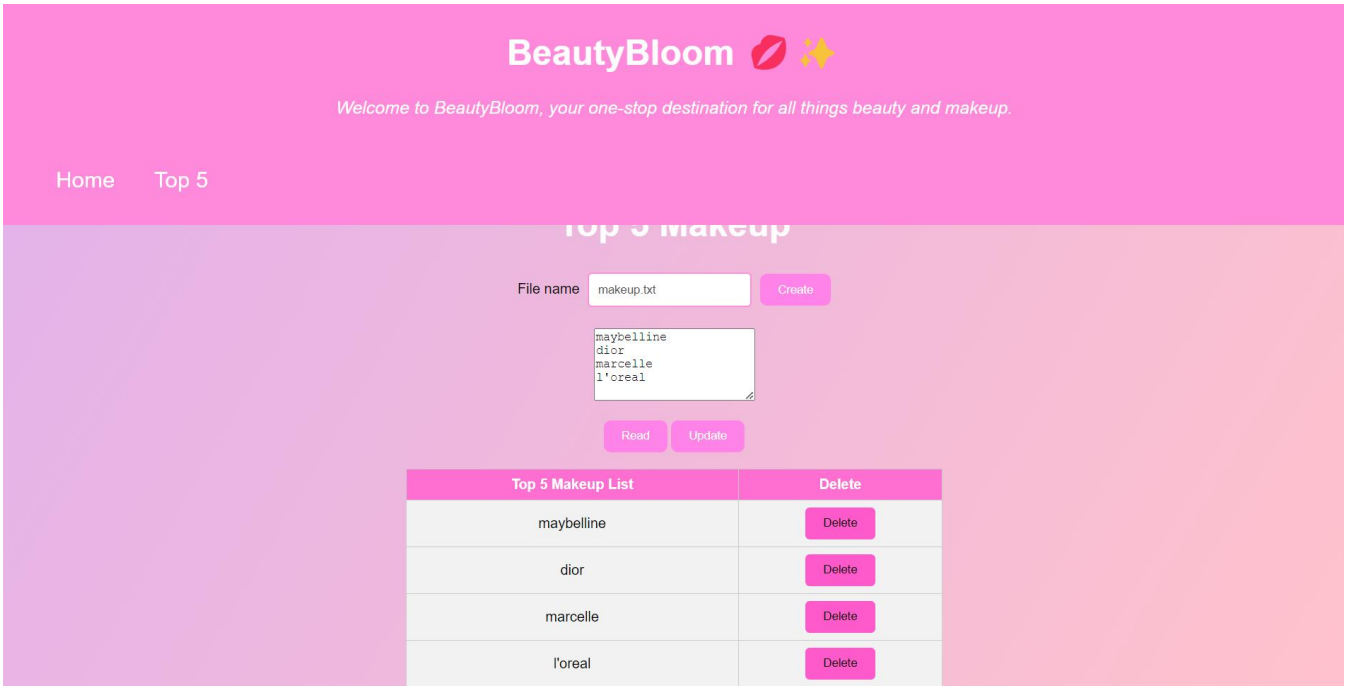


User clicks on the update button





User clicks on the delete button for certain makeup list



Coding for fetchapi.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
    <title>Homepage</title>
    <link rel="stylesheet" href="index.css" />
    <script src="fetchapi.js"></script> <!-- Include your JavaScript file -->
  </head>
  <body>
    <header>
      <center>
        <h1> BeautyBloom 💋✨</h1>
             <p>Welcome to BeautyBloom, your one-stop destination for all things beauty and
       makeup.</p>
      </center>
      <div class="navbar">
        <ul>
          <li><a href="fetchapi.html">Home</a></li>
          <li><a href="crud.html">Top 5</a></li>
        </ul>
      </div>
    </header>
    <br /><br /><br /><br /><br /><br /><br /><br /><br /><br /><br /><br /><br /><br />
    <center>
      <input type="text" id="searchData" placeholder="Enter product brand"> <!-- Input
field for brand -->
      <select id="searchData2"> <!-- Dropdown for product type -->
        <option value="">Select One</option>
        <option value="blush">Blush</option>
        <option value="bronzer">Bronzer</option>
        <option value="foundation">Foundation</option>
        <option value="eyebrow">Eyebrow</option>
        <option value="eyeliner">Eyeliner</option>
        <option value="eyeshadow">Eyeshadow</option>
        <option value="lipliner">Lipliner</option>
        <option value="lipstick">Lipstick</option>
        <option value="mascara">Mascara</option>
        <option value="nailpolish">Nail Polish</option>
      </select>
      <p id="productType"></p> <!-- Placeholder for product type display -->
      <button onclick="buttonClicked()">Search</button> <!-- Button to trigger the search
function -->
    </center>
    <ul id="productList"></ul> <!-- Placeholder for displaying product list -->
  </body>
</html>
```

Coding for fetchapi.js

```
// Function to handle the button click event
function buttonClicked() {
    // Get the brand and product type values from the input fields
    var brand = document.getElementById("searchData").value;
    var product_type = document.getElementById("searchData2").value;
    // Make an API request to fetch products based on brand and product type
    fetch(`http://makeup-
api.herokuapp.com/api/v1/products.json?brand=${brand}&product_type=${product_type}`)
        .then((response) => response.json())
        .then((data) => {
            console.log(data);
            // Check if any products were found
            if (data.length > 0) {
                displayProducts(data); // Call function to display products
            } else {
                // Display a message if no products were found
                document.getElementById("productList").innerHTML = "No products found for
the given brand and product type.";
            }
        })
        .catch((error) => {
            console.error("Error fetching data:", error);
            // Display an error message if there was an issue with the API request
            document.getElementById("productList").innerHTML = "Error fetching data.";
        });
}
// Function to display products in the HTML
function displayProducts(products) {
    const productList = document.getElementById("productList");
    productList.innerHTML = ''; // Clear previous product list

    // Iterate through the fetched products and display each product
    products.forEach(product => {
        const li = document.createElement('li');
        li.innerHTML = `<b>${product.name}</b><br><br> <b>Product Type:</b>
${product.product_type} <br><br> <b>Description:</b> ${product.description}<br><br>`;

// If an image link is available, create an image element and display the product image
        if (product.image_link) {
            const img = document.createElement('img');
            img.src = product.image_link;
            img.style.maxWidth = '200px'; // Adjust the image size if needed
            img.style.display = 'block'; // Set the image to be a block-level element
            img.style.margin = '0 auto'; // Center the image horizontally
            li.appendChild(img);
        }
        // Display the product price, or a message if the price is not available
        if (product.price) {
            li.innerHTML += `<br><br><center><b>Price:</b> ${product.price}
</center><br><br>`;
        } else {
            li.innerHTML += `<b>Price:</b> Price not available <br>`;
        }
        // If a product link is available, create a link to the product page
        if (product.product_link) {
            li.innerHTML += `<center><a href="${product.product_link}"
target="_blank">Product Link</a> <br></center>`;
        } else {
```

```
            li.innerHTML += `Product link not available <br>`;
        }
        li.innerHTML += '<br><br>'; // Add spacing between products
        productList.appendChild(li); // Append the product item to the product list
    });
}
```

Coding for crud.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8" />
<title>CRUD App</title>
<link rel="stylesheet" href="index.css"/> <!-- Include external CSS file -->
</head>
<body>
  <header>
    <center>
      <h1> BeautyBloom 💋✨</h1> <!-- Page title -->
      <p>Welcome to BeautyBloom, your one-stop destination for all things beauty and
makeup.</p> <!-- Page description -->
    </center>
    <div class="navbar">
      <ul>
        <li><a href="fetchapi.html">Home</a></li> <!-- Navigation link to Home page -->
        <li><a href="crud.html">Top 5</a></li> <!-- Navigation link to Top 5 page (current
page) -->
      </ul>
    </div>
  </header>
  <br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>
  <center>
    <div class="mainWrapper">
      <h1>Top 5 Makeup</h1> <!-- Main title for CRUD operations -->
      <form>
        <div class="form-group">
          <label>File name</label>
          <input id="fileName" type="text" class="form-control" value="makeup.txt"
disabled>
          <!-- Input field for file name (disabled, with a default value) -->
          <button id="btnCreate" class="btn btn-default">Create</button>
          <!-- Create button for creating files -->
        </div>
        <br>
        <textarea id="fileContents" class="form-control" rows="5"></textarea>
        <!-- Textarea for entering file contents -->
      </div>
    </form>
    <br>
    <button id="btnRead" class="btn btn-default">Read</button>
    <!-- Read button for reading file contents -->
    <button id="btnUpdate" class="btn btn-default">Update</button>
    <!-- Update button for updating file contents -->
  </div>
  <br><br>
  <table id="fileContents">
    <thead>
      <tr>
```

```
        <th>Top 5 Makeup List</th> <!-- Table header for product list -->
        <th>Delete</th> <!-- Table header for deletion of products -->
      </tr>
    </thead>
    <tbody id="fileTableBody"></tbody> <!-- Placeholder for displaying product list -->
  </table>
  </center>
  <script src="crud.js">
  </script>
</body>
</html>
```

Coding for crud.js

```
const { app, BrowserWindow } = require('electron');
const fs = require('fs')
const path = require('path')
const fileTableBody = document.getElementById('fileTableBody');

// Define event listeners and elements
var btnCreate = document.getElementById('btnCreate')
var btnRead = document.getElementById('btnRead')
var btnDelete = document.getElementById('btnDelete')
var btnUpdate = document.getElementById('btnUpdate')
var fileName = document.getElementById('fileName')
var fileContents = document.getElementById('fileContents')

let pathName = path.join(__dirname, 'Files')

// Event listener for the "Create" button
btnCreate.addEventListener('click', function(){ // Creating text file when the user clicks
the "Create" button
    let file = path.join(pathName, fileName.value)
    let contents = fileContents.value
    fs.writeFile(file, contents, function(err){
        if(err){
            return console.log(err)
        }
        var txtfile = document.getElementById("fileName").value
        alert(txtfile + " text file was created")
        console.log("The file was created")
    })
})
// Event listener for the "Read" button
btnRead.addEventListener('click', function() { // Reading contents of the created text file
    let file = path.join(pathName, fileName.value);

    // Read the file asynchronously
    fs.readFile(file, 'utf8', function(err, data) {
        if (err) {
            console.error(err);
            return console.log(err);
        }
        fileContents.value = data;

        // Split the file content into lines
        const lines = data.split('\n');

        const tableBody = document.getElementById('fileTableBody');
```

```
        tableBody.innerHTML = '';

        // Iterate through the lines and add them to the table
        lines.forEach(function(line) {
            addContentToTable(line);
        });

        console.log("Text file has been Read!");
    });
});
// Event listener to update data
btnUpdate.addEventListener('click', function () { // Updating the text file with new
contents
    let file = path.join(pathName, fileName.value);
    let contents = fileContents.value;

    fs.writeFile(file, contents, function (err) {
        if (err) {
            return console.log(err);
        }
        alert("File " + file + " has been updated")
        console.log("The file was updated!")
    });
})
// Function to add content to the table
function addContentToTable(content) {
    const newRow = document.createElement('tr');

    const contentCell = document.createElement('td');
    contentCell.textContent = content;

    const deleteCell = document.createElement('td');
    const deleteButton = document.createElement('button');
    deleteButton.textContent = 'Delete';

    // Event listener for the "Delete" button
    deleteButton.addEventListener('click', function() {
        const row = this.parentNode.parentNode;
        const rowIndex = row.rowIndex;

        row.remove();

        // Read the file, remove the specified line
        const file = path.join(pathName, fileName.value);
        fs.readFile(file, 'utf8', function(err, data) {
            if (err) {
                console.error(err);
                return console.log(err);
            }
            const lines = data.split('\n');
            lines.splice(rowIndex - 1, 1);
            const updatedData = lines.join('\n');
            fs.writeFile(file, updatedData, 'utf8', function(err) {
                if (err) {
                    console.error(err);
                }
            });
        });
    });
```

```
    deleteCell.appendChild(deleteButton);

    newRow.appendChild(contentCell);
    newRow.appendChild(deleteCell);

    const tableBody = document.getElementById('fileTableBody');
    tableBody.appendChild(newRow);
}
```

Coding for index.css

```
/* Body styles */
body {
  font-family: Arial, sans-serif;
  background: linear-gradient(120deg,rgb(225, 177, 236),rgb(255, 194, 204));
  margin: 0;
  padding: 0;
}
/* Style for the table */
table {
  width: 40%;
  border-collapse: collapse;
  border: 1px solid #ccc;
}
/* Style for table rows */
tr {
  background-color: #f2f2f2;
}
/* Style for table data cells */
td {
  padding: 8px;
  border: 1px solid #ccc;
  text-align: center;
}
/* Style for table header cells */
th {
  padding: 8px;
  border: 1px solid #ccc;
  text-align: left;
  background-color: #ff6ed1;
  color: #fff;
  text-align: center;
}
/* Style for the product list */
ul#productList {
  list-style: none;
  padding: 0;
  display: grid;
  grid-template-columns: repeat(3, 1fr); /* Three columns */
  gap: 10px; /* Gap between grid items */
}
/* Style for each product list item */
ul#productList li {
  background-color: #ffffff;
  padding: 15px;
  border-radius: 5px;
  box-shadow: 0 0 5px rgba(0, 0, 0, 0.2);
  max-width: 450px;
  margin-left: 20px;
  margin-right: 20px;
```

```
  margin-bottom: 20px;
  margin-top: 10px;
}
p {
  font-family: arial;
  color: white;
  font-style: italic;
  font-size: 20px;
}
/* Style for the "Search" button */
button {
  background-color: #ff5acb;
  color: #fff;
  padding: 10px 20px;
  border: none;
  border-radius: 5px;
  cursor: pointer;
  text-decoration: none; /* Remove underline */
  color: inherit; /* Inherit text color from its parent */
}
b {
  text-decoration: none; /* Remove underline */
  color: inherit; /* Inherit text color from its parent */
}
button:hover {
  background-color: #ff79d5;
}
/* Style for the button create */
#btnCreate {
  background-color: #ff83e8;
  color: #fff;
  padding: 10px 20px;
  border: none;
  border-radius: 8px;
  cursor: pointer;
}
/* Style for the button read */
#btnRead{
  background-color: #ff83e8;
  color: #fff;
  padding: 10px 20px;
  border: none;
  border-radius: 8px;
  cursor: pointer;
}
/* Style for the button delete */
#btnDelete{
  background-color: #ff83e8;
  color: #fff;
  padding: 10px 20px;
  border: none;
  border-radius: 8px;
  cursor: pointer;
}
/* Style for the button update */
#btnUpdate{
  background-color: #ff83e8;
  color: #fff;
  padding: 10px 20px;
  border: none;
```

```
  border-radius: 8px;
  cursor: pointer;
}
/* Style for the input field and dropdown */
input[type="text"],
select {
  padding: 10px;
  background-color: rgb(255, 255, 255);
  border: 2px solid #ff8edb;
  border-radius: 5px;
  margin: 5px;
}
.navbar{
  width: 100%;
  margin: auto;
  padding: 20px 0;
  display: flex;
  align-items: center;
  justify-content: space-between;
}
.navbar ul li{
  list-style: none;
  display: inline-block;
  margin: 0 20px;
  position: relative;
}
.navbar ul li a{
  text-decoration: none;
  color: #fff;
  font-size: 25px;
}
.navbar ul li::after{
  content: '';
  height: 4px;
  width: 0;
  background: rgb(239, 168, 251);
  position: absolute;
  left: 0;
  bottom: -10px;
  transition: 0.5s;
}
.navbar ul li:hover::after{
  width: 80%;
}
/* Style for the "Hello" heading */
h1 {
  color: #fefefe;
  font-style: bold;
  font-size: 40px;
}
/* Style for header */
header {
  position: fixed;
  top: 0;
  left: 0;
  right: 0;
  background-color: #ff89da;
  padding: 0;
  z-index: 100;
}
```

```
/* Style for error message */
#productList:empty::before {
  color: #797979;
  display: block;
  margin: 10px;
  position: center;
}
/* Style for API fetch error message */
#productList:empty::before {
  color: #999999;
  display: block;
  margin: 10px;
}
/* Popup styles */
.popup {
  display: none;
  position: fixed;
  top: 50%;
  left: 50%;
  transform: translate(-50%, -50%);
  z-index: 1000;
  background-color: #fff;
  padding: 20px;
  border: 1px solid #ccc;
  box-shadow: 0 0 10px rgba(0, 0, 0, 0.2);
}
/* Close button styles */
.close {
  position: absolute;
  top: 10px;
  right: 10px;
  cursor: pointer;
}
```

Coding for index.js

```
const { app, BrowserWindow } = require('electron');
const fs = require('fs')
const path = require('path')

// Handle creating/removing shortcuts on Windows when installing/uninstalling.
if (require('electron-squirrel-startup')) {
  // eslint-disable-line global-require
  app.quit();
}

const createWindow = () => {
  // Create the browser window.
  const mainWindow = new BrowserWindow({
    width: 800,
    height: 600,
    webPreferences: {
      nodeIntegration: true,
      contextIsolation: false,
    }

  });

  // and load the index.html of the app.
```

```
  mainWindow.loadFile(path.join(__dirname, 'fetchapi.html'));

  // Open the DevTools.
  //mainWindow.webContents.openDevTools();
};

// This method will be called when Electron has finished
// initialization and is ready to create browser windows.
// Some APIs can only be used after this event occurs.
app.on('ready', createWindow);

// Quit when all windows are closed, except on macOS. There, it's common
// for applications and their menu bar to stay active until the user quits
// explicitly with Cmd + Q.
app.on('window-all-closed', () => {
  if (process.platform !== 'darwin') {
    app.quit();
  }
});

app.on('activate', () => {
  // On OS X it's common to re-create a window in the app when the
  // dock icon is clicked and there are no other windows open.
  if (BrowserWindow.getAllWindows().length === 0) {
    createWindow();
  }
});

// In this file you can include the rest of your app's specific main process
// code. You can also put them in separate files and import them here.
```

Coding for preload.js

```
// See the Electron documentation for details on how to use preload scripts:
// https://www.electronjs.org/docs/latest/tutorial/process-model#preload-scripts
```

Link

https://github.com/ainzahirah/makeupbox.git